

AMENDMENTS TO THE CLAIMS

Please amend the claims of the present application as set forth below. In accordance with the PTO's revised amendment format, a detailed listing of all claims has been provided. A status identifier is provided for each claim in a parenthetical expression following each claim number. Changes to the claims are shown by strikethrough (for deleted matter) or underlining (for added matter).

1. (previously presented) A method of factoring operating system functions comprising:

defining criteria that governs how functions of an operating system are to be factored into one or more groups;

factoring the functions into one or more groups based upon the criteria; and associating groups of functions with programming objects that have data and methods, wherein the methods correspond to the operating system functions effective to provide an object oriented operating system, the programming objects being configured to be instantiated throughout a remote computing system.

2. (original) The method of claim 1, wherein the programming objects have interfaces through which the methods can be accessed.

3. (original) The method of claim 1, wherein the programming objects comprise COM objects.

1           4.     (original) The method of claim 1, wherein said factoring comprises  
2     creating a hierarchy of object interfaces in which certain interfaces can inherit  
3     from other interfaces.

4  
5           5.     (original) The method of claim 1, wherein said factoring comprises  
6     creating a hierarchy of object interfaces in which certain interfaces can aggregate  
7     with other interfaces.

8  
9           6.     (original) The method of claim 1 further comprising instantiating a  
10    plurality of programming objects across a process boundary.

11  
12          7.     (original) The method of claim 1, further comprising instantiating a  
13    plurality of programming objects across a machine boundary.

14  
15          8.     (original) The method of claim 1, wherein the criteria is based, at  
16    least in part, on the manner in which particular functions behave.

17  
18          9.     (original) The method of claim 8, wherein the manner includes a  
19    consideration of the types of operating system resources that are associated with  
20    the operation of a function.

21  
22          10.    (original) The method of claim 8, wherein the manner includes a  
23    consideration of whether a particular function creates an operating system  
24    resource.

25

11. (original) The method of claim 8, wherein the manner includes a consideration of whether a particular function operates upon an operating system resource.

12. (original) The method of claim 1, wherein the criteria is based, at least in part, on the manner in which particular functions behave, wherein the manner includes:

a consideration of the types of operating system resources that are associated with the operation of a function; and

a consideration of whether a particular function creates an operating system resource.

13. (original) The method of claim 1, wherein the criteria is based, at least in part, on the manner in which particular functions behave, wherein the manner includes:

a consideration of the types of operating system resources that are associated with the operation of a function call; and

a consideration of whether a particular function operates upon a given operating system resource.

14. (previously presented) A method of factoring operating system functions comprising:

factoring a plurality of operating system functions that are used in connection with operating system resources into first groups based upon first criteria;

1 factoring the first groups into individual sub-groups based upon second  
2 criteria; and

3 assigning each sub-group to its own programming object interface, wherein  
4 a programming object interface represents a particular object's implementation of  
5 its collective methods effective to provide an object-oriented operating system,  
6 wherein individual objects having associated programming object interfaces are  
7 configured to be instantiated throughout a remote computing system.

8  
9 15. (original) The method of claim 14, wherein the first criteria is based  
10 upon the type of resource that is associated with an operation of a function.

11  
12 16. (original) The method of claim 14, wherein the second criteria is  
13 based upon the nature of an operation of a function on a particular resource.

14  
15 17. (original) The method of claim 16, wherein said nature concerns  
16 whether a function creates a resource.

17  
18 18. (original) The method of claim 16, wherein said nature concerns  
19 whether a function does not create a resource.

20  
21 19. (original) The method of claim 14, wherein the first criteria is based  
22 upon the type of resource that is associated with an operation of a function, and the  
23 second criteria is based upon the nature of an operation of a function on a  
24 particular resource.  
25

1           20. (original) The method of claim 14, wherein at least one interface  
2 inherits from another interface.

3  
4           21. (original) The method of claim 14, wherein at least one interface  
5 aggregates with another interface.

6  
7           22. (original) The method of claim 14 further comprising instantiating a  
8 plurality of programming objects across a process boundary.

9  
10           23. (original) The method of claim 14 further comprising instantiating a  
11 plurality of programming objects across a process boundary and a machine  
12 boundary.

13  
14           24. (previously presented) A method of factoring operating system  
15 functions comprising:

16           factoring a plurality of operating system functions into interface groups  
17 based upon the resources with which a function is associated;

18           factoring the interface groups into interface sub-groups based upon each  
19 function's use of a handle that represents a resource; and

20           organizing the interface sub-groups so that at least one of the interface sub-  
21 groups inherits from at least one other of the interface sub-groups, individual  
22 interface sub-groups being associated with individual programming objects that  
23 can be instantiated throughout a remote computing system.

24

25

1           25. (original) The method of claim 24, wherein said organizing  
2 comprises aggregating at least one of the interface sub-groups.

3  
4           26. (original) The method of claim 24, wherein the interface sub-groups  
5 are associated with COM objects.

6  
7           27. (original) The method of claim 24, wherein the factoring of the  
8 interface groups into interface sub-groups comprises considering whether a  
9 function creates a handle.

10  
11           28. (original) The method of claim 24, wherein said organizing  
12 comprises aggregating at least one of the interface sub-groups, and wherein the  
13 factoring of the interface groups into interface sub-groups comprises considering  
14 whether a function call creates a handle.

15  
16           29. (currently amended) An operating system application program  
17 interface embodied on a computer-readable medium comprising a plurality of  
18 object interfaces, wherein each object interface is associated with an object that  
19 includes one or more methods that are associated with and can call functions of an  
20 operating system that does not comprise the object interfaces, individual at least  
21 one said objects being configured to be remotely instantiated ~~in-process, locally, or~~  
22 ~~remotely~~.

1 30. (original) The operating system application program interface of  
2 claim 29, wherein the object interfaces are arranged in groups in accordance with  
3 the types of objects with which their operation is associated.  
4

5 31. (original) The operating system application program interface of  
6 claim 29, wherein the methods within some of the interfaces are arranged in  
7 accordance with whether they create an object.  
8

9 32. (original) The operating system application program interface of  
10 claim 29, wherein the methods within some of the interfaces are arranged in  
11 accordance with whether they do not create an object.  
12

13 33. (original) The operating system application program interface of  
14 claim 29, wherein the methods within some of the interfaces are arranged in  
15 accordance with whether they operate upon an object.  
16

17 34. (original) The operating system application program interface of  
18 claim 29, wherein at least some of the object interfaces are arranged so that they  
19 inherit from other of the object interfaces.  
20

21 35. (original) The operating system application program interface of  
22 claim 29, wherein at least some of the object interfaces are arranged so that they  
23 aggregate with other of the object interfaces.  
24

25 36. (previously presented) An operating system comprising:

1 a plurality of programming objects having interfaces, wherein the  
2 programming objects represent operating system resources, and wherein the  
3 interfaces define methods that are organized in accordance with whether they  
4 create an operating system resource or not;

5 wherein the programming objects are configured to be called either directly  
6 or indirectly by an application; and

7 wherein the methods are configured to call operating system functions  
8 responsive to being called directly or indirectly by an application;

9 said programming objects being configured to be instantiated throughout a  
10 remote computing system.

11  
12 37. (original) The operating system of claim 36, wherein some of the  
13 objects are disposed across at least one process boundary.

14  
15 38. (original) The operating system claim 36, wherein some of the  
16 objects are disposed across at least one machine boundary.

17  
18 39. (previously presented) The operating system of claim 36, wherein at  
19 least some of the objects are disposed across at least one process boundary and at  
20 least one machine boundary.

21  
22 40. (previously presented) The operating system of claim 36, wherein  
23 the objects comprise COM objects.

24

25



1           41. (previously presented) A method of converting an operating system  
2 from a non-object-oriented format to an object oriented format, wherein the  
3 operating system includes a plurality of operating system functions that are  
4 callable to create or use operating system resources, the method comprising:

5           defining a plurality of programming object interfaces that define methods  
6 that correspond to the operating system functions, wherein programming objects  
7 that support the interfaces are callable either directly by an application that makes  
8 object-oriented calls, or indirectly by an application that makes function calls, said  
9 programming objects being configured to be instantiated throughout a remote  
10 computing system;

11           calling a programming object interface either directly via an object-oriented  
12 call, or indirectly via an indirection that transforms a function call into an object-  
13 oriented call; and

14           responsive to said calling, calling an operating system function with a  
15 method of the programming object that supports said programming object  
16 interface.